Incorporating Temporal Common Sense in a Conversational Chatbot

A Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Pranav Guruprasad Rao & Rachit Sandeep Jain
(112101038 & 112101019)

INDIAN INSTITUTE
OF TECHNOLOGY
PALAKKAD

COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

CERTIFICATE

*This is to certify that the work contained in the project entitled "**Incorporating Temporal Common Sense in a Conversational Chatbot**" is a bonafide work of **Pranav Guruprasad Rao & Rachit Sandeep Jain** (**Roll No. 112101038 & 112101019**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Palakkad under my guidance and that it has not been submitted elsewhere for a degree.*

Dr Koninika Pal

Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Palakkad

Acknowledgements

We wish to express our sincere gratitude to our guide **Dr Koninika Pal** for her constant support and guidance throughout the project.

Pranav Rao & Rachit Jain

112101038 & 112101019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this project, we shall be creating a chatbot which is able to grasp temporal information and commonsense from the conversations.

Contemporary chatbot systems often exhibit limited contextual understanding, particularly in interpreting temporal data. They tend to provide shallow responses to prompts or conversations, lacking depth in contextual comprehension. Instances arise where these chatbots struggle to infer meanings from temporal references, such as discerning time disparities between events or comprehending past occurrences and their implications.

For example, taking two events as "going to have a vacation" and "going out for a walk," most people stick to their minds realizing that a vacation is generally longer and is taken less frequently than a walk. and even though current development in this area have been promising, there is still a long way to go for computers to internalize temporal commonsense.

The current implementation of the project can be found at the following Github link.

Temporal Sense refers to the practical knowledge, intuition, or knowledge in depth got as a result of temporal knowledge of the world. These are such as temporal context of phrases, actions and events.

e.g. Temporal reasoning means a chatbot is indeed capable of recognising temporal properties such as the order of events, duration of events, size of the time difference, and so on.

Described temporals or temporal expressions in text is a set of tokens which represents a period of time. Examples of a temporal frame are:

- Time point (6 May 1980, Monday, 12:00 PM)
- Time span (7 minutes, 5 years, 2 months)

There are five temporal properties:

1. Duration (how long an event takes)
2. Temporal Ordering (typical order of events)
3. Typical Time (when an event happens)
4. Frequency (how often an event occurs)
5. Stationarity (whether a state holds for a very long time or indefinitely)

Therefore, Temporal Common Sense is applied in temporal natural language processing/inference.

In this project, we roughly attempt to assess to what extent these various models represent the relative positions that different points in time may occupy concerning the other and the relative durations of those points. We also consider whether these models can

reason in terms of durations quantised at different units. This project attempts at solving the Limited Temporal Understanding and the Shallow depth in contextual understanding through the design of feasible logics and models for understanding temporal characteristics of textual inputs.

1.1 Problem Statement

The primary objective is to develop logical schemas enabling the chatbot to leverage or approximate temporal common sense, thereby enhancing its contextual understanding. Subsequently, we aim to construct a prototype chatbot incorporating these logical schemas.

Here is the mathematical formulation of the problem statement

Consider a given context C_{curr}, which is the set of sentences representing the current context:

$$C_{\text{curr}} = \{s_1, s_2, \ldots, s_m\}$$

where $s_1, s_2, \ldots, s_m$ are the sentences in the context. Our task is to predict the next sentence s_{next}.

The first subproblem is to classify the type of the next sentence s_{next}, i.e., whether it belongs to the temporal category or not. We define the possible categories for s_{next} as:

$$\{s_{\text{temp}}, s_{\text{other}}\}$$

where s_{temp} refers to temporal sentences, and s_{other} refers to non-temporal sentences.

Our goal is to train a classifier $\mathcal{C}$ that predicts the category of s_{next} based on the current context C_{curr}:

$$\mathcal{C} : f(C_{\text{curr}}) \rightarrow \{s_{\text{temp}}, s_{\text{other}}\}$$

where $f(C_{\text{curr}})$ is a function that represents the feature extraction process, capturing relevant signals from the context and any supplementary information to make the prediction.

The second subproblem involves generating s_{next} based on the predicted category.

- If s_{next} is classified as temporal (s_{temp}), then we generate s_{next} using a temporal model M_{temp}, which references both the current context C_{curr} and a timeline T to produce the next temporal sentence:

$$s_{\text{next}} = M_{\text{temp}}(C_{\text{curr}}, T)$$

- If s_{next} is classified as non-temporal (s_{other}), then we generate s_{next} using a non-temporal model M_{other}, based solely on the current context:

$$s_{\text{next}} = M_{\text{other}}(C_{\text{curr}})$$

Thus, the generation of s_{next} depends on whether it is temporal or non-temporal, and the corresponding model uses either the timeline or just the current context.

1.2 Organization of The Report

This chapter provides an introductory knowledge for the project. In the upcoming chapters, we shall be elaborating on our methods, our prior work and literature surveys.

We first explain on our prior work on this project in Chapter 2. We then explain the existing literature on this topic and the different datasets that we shall be using in Chapter 3. We then explain our approach to the problem in Chapter 3 & 4. We then conclude with Future work.

Chapter 2

Literature Review

2.1 Datasets used

Here are some of the datasets that we have used till now.

1. TIMEDIAL
2. BIGBench Temporal Ordering -
    - Each sample in the dataset has a set of sentences which narrate certain events that occured in a span of a day with their event duration; the dataset is used for both event duration and temporal ordering

In this section, we summarize key papers and approaches relevant to temporal commonsense reasoning and event ordering.

2.1.1 Do Language Models Have a Common Sense regarding Time? Revisiting Temporal Commonsense Reasoning in the Era of Large Language Models

This paper evaluates the temporal commonsense reasoning capabilities of large language models (LLMs). It focuses on:

- The performance of LLMs on various temporal reasoning tasks such as event ordering, event duration, and frequency.
- The challenges LLMs face, such as reasoning about multiple events over long time frames and handling ambiguous temporal expressions.
- The paper highlights strong performance of models like GPT-3.5 and FLAN-T5, particularly in event frequency and duration tasks, while they struggle with specific event timings and understanding complex temporal states.

2.1.2 LTLBench: Towards Benchmarks for Evaluating Temporal Logic Reasoning in Large Language Models

This paper introduces LTLBench, a benchmark designed for evaluating temporal logic reasoning (TR) in LLMs. Key contributions include:

- A novel pipeline for constructing datasets that test LLMs' ability to handle complex temporal reasoning problems using Linear Temporal Logic (LTL).
- The use of directed graphs and randomly generated LTL formulas to create scenarios for testing LLMs on event transitions and temporal relations.
- Strong performance by LLMs on simple TR problems but noticeable challenges when handling more complex LTL operators.

2.1.3 Once Upon a Time in Graph: Relative-Time Pretraining for Complex Temporal Reasoning

This paper explores the use of graph-based models for improving temporal reasoning in LLMs. It introduces:

- Relative-time pretraining as a method for enhancing the ability of LLMs to understand complex temporal relationships between events.

- A model called TaCOMET, which augments commonsense knowledge with temporal relations between events to handle time-aware reasoning more effectively.

2.1.4 Toward Building a Language Model for Understanding Temporal Commonsense

This paper proposes a language model designed to enhance understanding of temporal commonsense, using multi-step fine-tuning and multi-task learning approaches:

- The model leverages continual pre-training with tasks like Sentence Order Prediction (SOP) and Masked Language Modeling (MLM) to improve temporal reasoning abilities.

- Results indicate improvements in tasks such as event ordering, timing, and temporal duration, achieving strong performance with pre-trained models like BERT, RoBERTa, and ALBERT.

Chapter 3

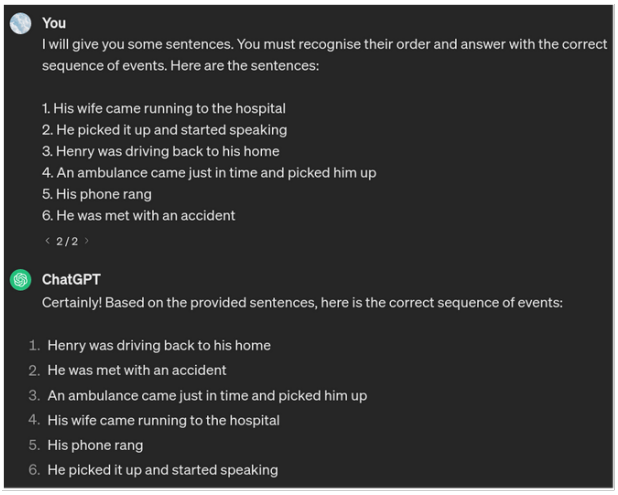
Review of Prior Works

This project extends our previous Open Ended Lab project under the same mentor. Here we review our previous approaches.

3.1 Problems in Present Chatbots

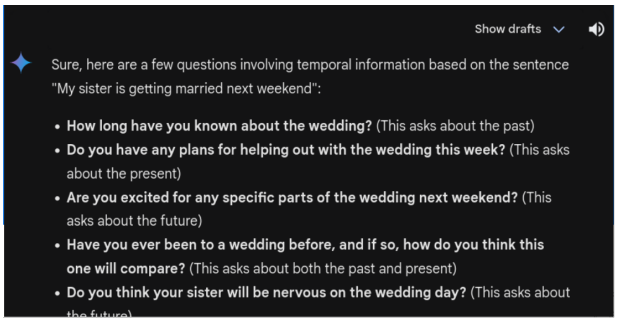
Current chatbots like ChatGPT and Gemini struggle with temporal reasoning. We conducted a survey to test this, and here are the results.

3.1.1 ChatGPT 3.5/4



ChatGPT successfully arranged a set of jumbled sentences in the correct order.

3.1.2 Gemini



Gemini was asked to generate various temporal questions, as shown above.

3.2 Dataset

We utilized the TIMEDIAL dataset and split it into training and testing sets.

Training Dialogues	1395
Testing Dialogues	50
Training Data Size	27132
Testing Data Size	1268
Avg Rows per Dialogue	19

Table 3.1 Dataset Information

Each dialogue is split into consecutive subdialogues to increase the dataset size. For example:

Original Dialogue:

1. A: We need to take the accounts system offline...
2. B: How long will it be down?
3. A: It will be down for at least 12 hours.

Split Dialogues:

1. Set 1: (Dialogue 1 only)
2. Set 2: (Dialogue 1 and 2)
3. Set 3: (All dialogues)

3.2.1 Dataset Columns

1. Input: List of dialogues with temporal flags
2. Output: A sentence with or without temporal context
3. Chunk: A set of sentences

3.3 Model

We used T5 (Text-to-Text Transfer Transformer) as the base model, which is well-suited for text generation and classification tasks. T5 includes:

1. Encoder-Decoder architecture
2. Pre-training on fill-in-the-blank tasks

3.3.1 Hyperparameters

Key hyperparameters include:

1. Learning Rate: Governs the pace of updates
2. Dropout Rate: Prevents overfitting
3. Epochs: Number of training iterations
4. Batch Size: Number of samples per iteration

3.4 Process

3.4.1 Identifying Temporal Words

We used the Timexy module to identify and categorize temporal words using TIMEX3 encoding.

3.4.2 Preprocessing and Procedure

Preprocessing includes tokenizing sentences and passing them through the model. Key steps:

1. Randomizing training data
2. Setting hyperparameters
3. Initializing the model and tokenizer
4. Training the model and testing

3.4.3 Cross Validation

We used k-Fold Cross Validation to find the optimal values for:

1. Learning Rate
2. Epochs
3. Dropout Rate

3.5 Results

We experimented with different hyperparameters. Here are some results:

Learning Rate	Epochs	Dropout Rate	Train Loss	Rouge Sum
3.00E-05	6	0.01	0.0152	0.416
3.00E-04	4	0.6	0.0125	0.577
3.00E-05	5	0.5	0.0411	0.650

Table 3.2 Training Results

Learning Rate	Epochs	Dropout Rate	Rouge Sum
3.00E-04	5	0.5	0.382
3.00E-04	4	0.6	0.577

Table 3.3 Testing Results

3.5.1 Key Takeaways

We identified the limitations of present chatbots in temporal reasoning and developed a T5-based chatbot for temporal tasks. The main approach we took here was to fine-tune a model according to the TIMEDIAL dataset. However, our model faced challenges with long outputs and limited dataset size and the results can be further improved. The next steps to look into were to increase the dataset size, and improving on the overall model structure.

Chapter 4

Model Structure

After going through our previous solution as well as the present literature, we propose an advanced solution for the same.

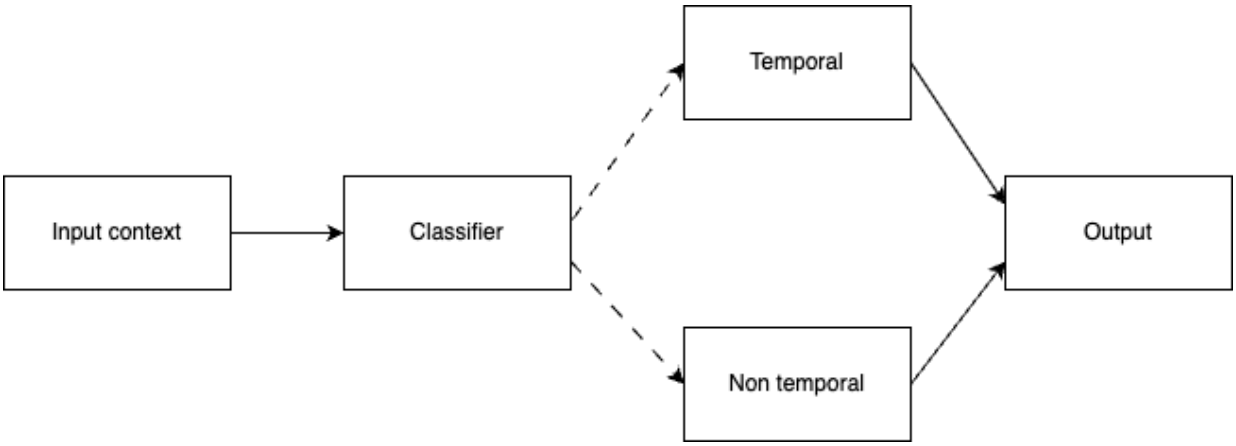


Fig. 4.1 Model Architecture

- **Input Context:** The input to the model, the actual data that is given, in the form of a sentence or a conversation. It can also contain temporal or non temporal data. The model first receives this, accepts an input and transforms it in order to decide the right path that the output should follow.

- **Classifier:** The classifier examines the surrounding context in relation to the input and determines whether or not the temporal dependent **output** is needed. If

the output is to be time-oriented, it sends the context to be processed through the **Temporal Pathway**. Otherwise, the context is processed via the **Non-Temporal Pathway**.

- **Temporal Model:** In the case where the classifier reached a conclusion that the output requires temporal reasoning, the input context is passed to the **Temporal Model** as shown in Figure 5.2. Here, we incorporate event extraction and prediction along with a timeline to craft the response.
- **Non-Temporal Model:** When the classifier decides that the output does not require temporal reasoning, the input is passed to the **Non-Temporal Model**, which handles general commonsense reasoning without involving any temporal dependencies.
- **Output:** The final output is produced after processing the input context via the appropriate model (temporal or non-temporal), based on the classifier's decision.

4.1 Classifier

The model is designed to categorize the output as temporal or non temporal based on the input context. The classifier is to determine whether the temporal commonsense reasoning in output should be involved or not. If the model refers to temporal, then a timeline mechanism is used in order to produce the temporal output. Otherwise, it follows a non-time based route.

4.1.1 Data Preparation

The training data comes from the `TimeDial` dataset, which has been modified to contain two columns: an `Input` column for context and an `Output` column for text. Each output sentence was further converted into binary labels (0 or 1), where a label of 1 indicates that the sentence contains temporal information, and a label of 0 denotes that it does not.

4.1.2 Temporal Label Generation

To generate the temporal labels, `spaCy` was initially used for detecting temporal cues within the output texts. `spaCy` is effective at identifying named entities such as specific dates, durations, or times. However, it does not capture certain temporal adverbs and question forms crucial for temporal reasoning, such as "when," "how often," and "how long."

To overcome this limitation, `spaCy` was augmented with rule-based conditions. The added rules capture:

- Temporal adverbs such as *when*, *how often*, and *how long*.
- Prepositions related to time like *during* and *at*.
- Common time markers like *after*, *before*, *tomorrow*, and *yesterday*.
- Auxiliary verbs like *will*, *is*, and *are*, which often signal temporal relations.

By combining `spaCy`'s named entity recognition with these rules, the temporal label generation process was able to provide more accurate labels for training the classifier.

4.1.3 Classifier Model

The classifier represents the center of the model and employs `BERT` (Bidirectional Encoder Representations for Transformer). from Transformers with an extra fully connected layer plus a dropout layer that helps with the overfitting issue. The model is fine-tuned for sequence classification, and the training process uses a threshold-based approach for binary classification. The threshold, initially set to 0.55, determines whether the predicted probability indicates a temporal or non-temporal label.

Training Process

The dataset was split in a 90:10 ratio for training and testing. The training set was further divided into 80% for training and 20% for validation. The model was trained over 5 epochs using a dropout rate of 0.02, a learning rate of $2e^{-5}$, and a threshold of 0.55. `CrossEntropyLoss` was used as the loss function for classification.

The training procedure includes a scheduler with a linear warm-up for learning rate adjustment, helping the model to converge effectively.

By building on robust preprocessing and a well-defined classification framework, this classifier lays the groundwork for distinguishing temporal and non-temporal contexts. Its ability to accurately categorize input ensures the effectiveness of subsequent modules, such as the temporal and non-temporal reasoning paths.

4.2 Temporal Model

The Temporal Model is central to enabling our chat bot to reason about time-sensitive contexts. The Model has two components: an Encoder and a Decoder. The Encoder predicts the next temporal token using event extraction and the Decoder will generate the response suitable for the current context referring to the timeline.

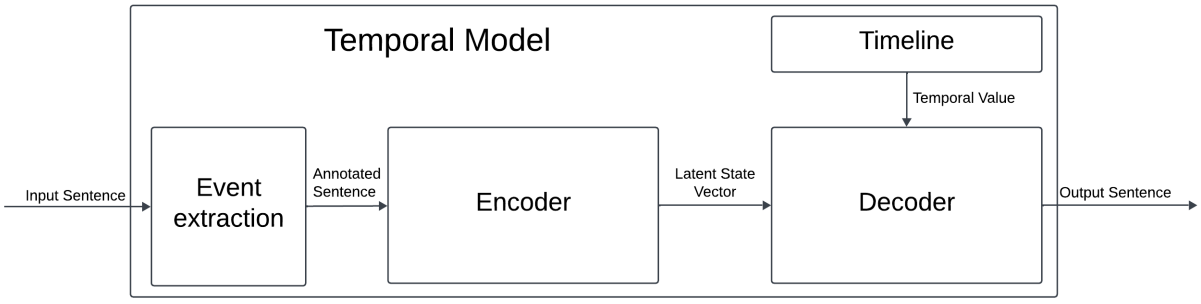


Fig. 4.2 Temporal Model Structure

The Temporal Model is trained in two phases:

1. **Encoder Training:** A BERT-based encoder is trained using masked datasets

2. **Decoder Training:** Once the encoder is trained, a decoder is trained to generate time-aware responses using the learned representations from the encoder.

4.2.1 A short description of Generative Models

Generative Models learn the prior probability distribution of the data points to be generated i.e $P(x)$. This is done by assuming a latent variable z which has a causal relationship with x. This forms the **Latent Variable Model** where we learn a mapping from z to x:

$$p(x) = \int p(x|z)p(z)dz \tag{4.1}$$

where $p(x|z) = D(z)$ is learnt by a neural network and $p(z)$ is known. The issue is the $p(x)$ is intractable.

To get around this problem, we use the Bayes' Theorem,

$$p(x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(z|x)} \tag{4.2}$$

The Function $p_\theta(z|x) = E(x)$ forms the **Encoder** while the $p_\theta(x|z) = D(z)$ forms the **Decoder**.

The Encoder takes in a data point x and finds the latent distribution z corresponding to it. This latent distribution is used by the Decoder to sample a fresh value x' as the output.

4.2.2 Loss Function

With Neural Networks comes the definition of Loss Functions. There shall be two loss functions: one for Encoder and one for the Decoder. The Encoder's loss function deals with the proper recognition or approximation of the $E(z)$ with respect to $p(z)$ and the

Decoder's loss function deals with the generation of the fresh data point from the hidden state.

Let L_e be the loss function of the Encoder and L_d be the loss function of the Decoder. Let $z, \hat{z}$ be the actual and predicted latent distributions respectively, and x and $\hat{x}$ be the actual and predicted data values. Then the final Loss function L_f is given by:

$$L_f = \lambda_e L_e(z, \hat{z}) + \lambda_d L_d(x, \hat{x}) \tag{4.3}$$

where λ_1, λ_2 are weights with the following conditions: $0 \leq \lambda_1, \lambda_2 \leq 1$ and $\lambda_1 + \lambda_2 = 1$.

4.2.3 Encoder

An encoder takes in a sentence and predicts the next temporal token. It does this by learning to associate different temporal aspects together. In essence, the encode learns the hidden distribution behind the temporal sentences, and provides a hidden state in the form of vector embeddings which encodes our temporal information.

4.2.4 Event Extraction

It will also be seen that event extraction is a key enabler for making the Temporal Model to reason about temporal contexts. The main goal of event extraction is to catalogue and identify temporal information from sentences and this creates the basis for interpret knowledge and use reasoning to understand temporal relationship conversations.

The main objectives of event extraction are:

- Identifying **key events** or actions described in a sentence.
- Extracting **associated timepoints** (e.g., "6 PM," "yesterday").
- Parsing **durations** (e.g., "12 hours," "two weeks").

This extracted information ensures that the model has a structured understanding of events and their temporal attributes, which is crucial for generating coherent and contextually accurate responses in temporal scenarios.

Tools and Techniques

The event extraction pipeline combines multiple tools and methods to ensure accurate extraction:

- **spaCy**: Used for parsing sentence structure and identifying temporal expressions like dates and durations.
- **Heideltime**: Detects relative temporal expressions (e.g., "yesterday," "next week") and converts them into standardized timestamps.
- **Custom Rules**: Handles specific temporal patterns that may not be captured by automated tools, such as:
  - Temporal adverbs (*when, how long, how often*).
  - Prepositions related to time (*during, at, after*).
  - Common markers (*yesterday, tomorrow, next month*).

These techniques collectively enable the extraction of precise and comprehensive temporal information from dialogues.

Process Overview

The event extraction process involves:

1. Parsing each sentence in the dialogue to detect events and their temporal attributes.
2. Using **spaCy** and **Heideltime** to annotate temporal elements, such as event occurrence times and durations.

3. Applying custom rules to ensure coverage of missing or ambiguous temporal expressions.

By systematically extracting and organizing temporal data, the process provides the Temporal Model with the essential information required for understanding and reasoning about time-based contexts.

Temporal Graph

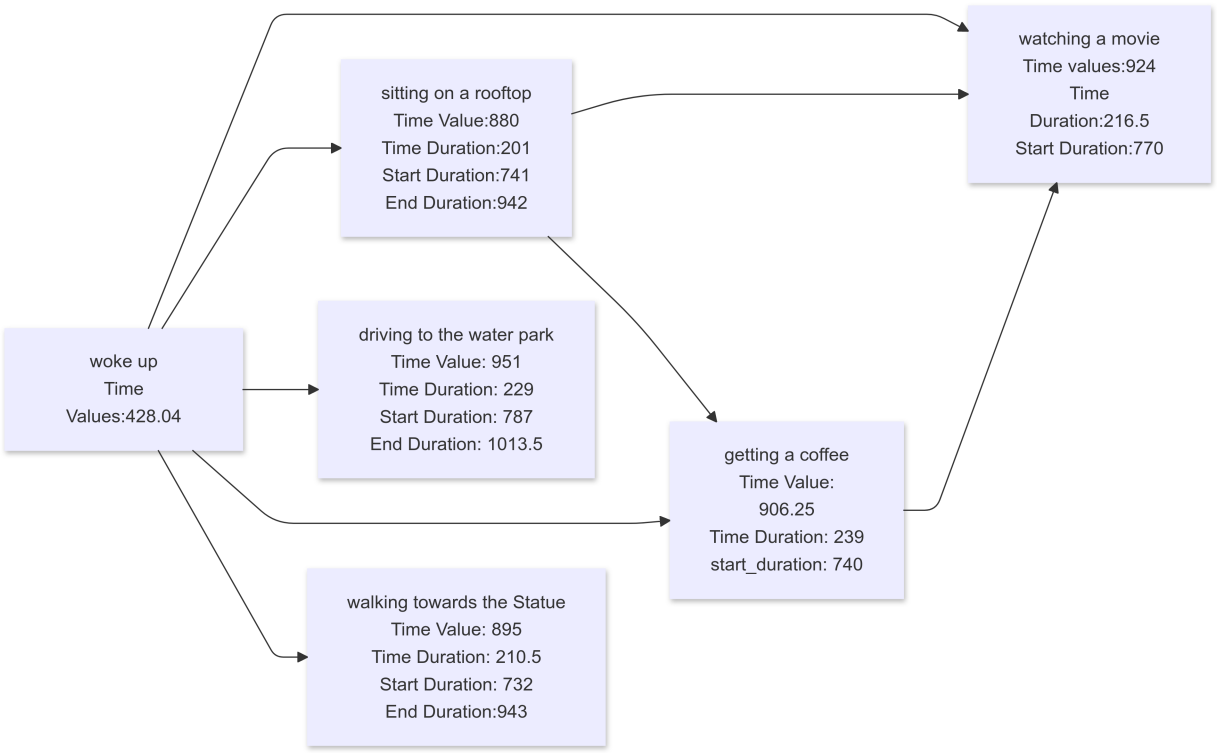


Fig. 4.3 A simplified subgraph of the overall Temporal Graph

After we extract events, we have to link them together. The events will link in a graph structure, with each event being affected by various other events and other events leading to it.

The Graph node has the following attributes:

- current event
- neighbors (next events)
- time point of occurrence
  - exact time value of the event
  - time duration of the event (start, end, duration)

An edge exists between two nodes in a graph when they occur consecutively or are semantically linked.

We accumulate each graph node into a dataset which we shall use for training the temporal model. For each of the time values present in the node, we mention the average time point and its approximate minimum and maximum time. To find the approximate minimum and maximum time values, we find the range of values which are at one standard deviation distance from the average and take their minimum and maximum values.

Graph Node Example:

- **current event** : getting a coffee
- **ngbs** : ['watching a movie', 'sitting on a rooftop', 'driving to the water park', ...]
- **time point** :
  - **time value** : [610.0, 876.4640883977901, 1159.0]
  - **time duration** : [61, 216.68098159509202, 427]
  - **start duration** : [488, 734.9938650306749, 976]
  - **end duration** : [671, 947.9325153374233, 1220]

The temporal graph ensures that the Temporal Model reasons about events in relation to their sequence, duration, and context.

Model Training

We fine-tuned BERT-like models for the encoder, namely BERT-base, RoBERTa, and distilBERT, using the Masked Language Modeling (MLM) technique where we mask out some information and let the model predict it.

We used the Temporal Ordering dataset and created a temporal graph of the same. In order to further reinforce the time value predictions, we also include the following values for both time point and time duration:

- **minimum value**
- **average value**
- **maximum value**

Some statistics about the Temporal Graph:

- No of nodes: 26
- Average no of edges: 21
- Density: 1.6553

Additionally, we introduce custom tokens such as $\langle EVENT \rangle$, $\langle TIME\ DURATION \rangle$ and $\langle TIME\ VALUE \rangle$ to guide the model towards more specific associations.

Thus, We generate a set of masked sentences from a single sentence in which only one temporal token is masked.

Example of the Masked Dataset: The encoder receives a prompt with masked attributes, requiring it to predict the masked token.

- **System Prompt:** I will provide a sentence containing details about the current event, its next event, its occurrence time, and its duration. One of these attributes will be masked, and your task is to predict the masked token.
- **Example Input:**
  - **Sentence:**

    Current Event: attending class; Minimum Time point: 610.0 minutes; Average Time point: 882.3680981595093 minutes; Maximum Time point: 1159.0 minutes; Minimum Time duration: 61 minutes; Average Time duration: 202.63698630136986 minutes; Maximum Time duration: 366 minutes; Minimum Start time: [MASK] minutes; Average Start time: 750.3835616438356 minutes; Maximum Start time: 1037 minutes; Minimum End time: 671 minutes; Average End time: 948.8424657534247 minutes; Maximum End time: 1220 minutes; Next event: waiting at the airport.
  - **Mask Value:** 488

The Masked dataset has 7532 sentences.

Some training details are as follows:

- Optimizer: ADAM
- Learning Rate: $3e^{-5}$
- Epoch size: 3
- Weight Decay: 0.01
- Batch size: 16

4.2.5 Decoder

The decoder is responsible for generating coherent and temporally-aware sentences based on the contextual representations learned by the encoder. It takes as input the embeddings produced by the encoder and constructs a complete response that incorporates temporal reasoning.

The decoder leverages the timeline and temporal graph information to integrate knowledge about events, their order, and durations. It ensures that the output respects the temporal relationships between events, as determined by the encoder's predictions.

The training process for the decoder involves pairing masked inputs with their expected outputs. This process enhances the model's ability to reconstruct and predict temporal features in real-world scenarios.

Timeline

Here we will discuss the concept of incorporating a Timeline for reference to the model.

The intuition behind this is that the way humans can reason regarding event duration, stationarity and typical time of events is through experience - we have seen the events occurring at their particular time periods for a long time. This common knowledge about events has to be incorporated into LLMs in order for them to reason on their own. The timeline which we build collects events and their occurring periods and maps events to each other based on the dialogues present in the dataset.

4.3 Non-Temporal Model

For now, we aim to focus on the temporal model before moving into the non-temporal model. We hypothesise that the non-temporal model will handle sentences/questions which

do not inherently involve time as a context.

Chapter 5

Results

5.1 Classifier Results

After training, the classifier model was evaluated on the test set. The following performance metrics highlight its effectiveness:

- **Testing Accuracy:** 88.52%
- **Testing Precision:** 88.48%
- **Testing F1 Score:** 88.42%

5.1.1 Training and Validation Results

The training process yielded strong validation results:

- **Validation Accuracy:** 85.43%
- **Validation Precision:** 85.31%
- **Validation F1 Score:** 85.30%

5.1.2 Plots

The training loss and validation accuracy during the training process are shown in Figure 5.1.

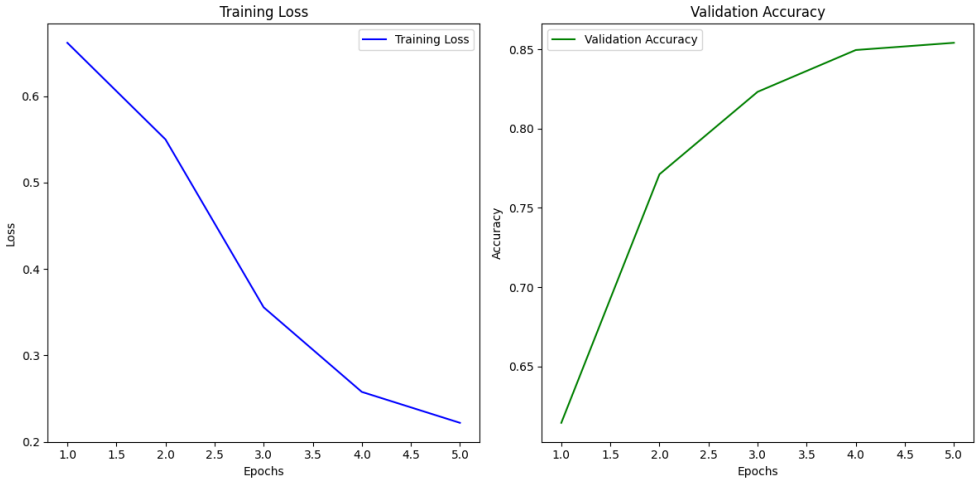


Fig. 5.1 Training Loss and Validation Accuracy over Epochs for Classifier

5.2 Encoder Results

The Encoder's training results are as follows:

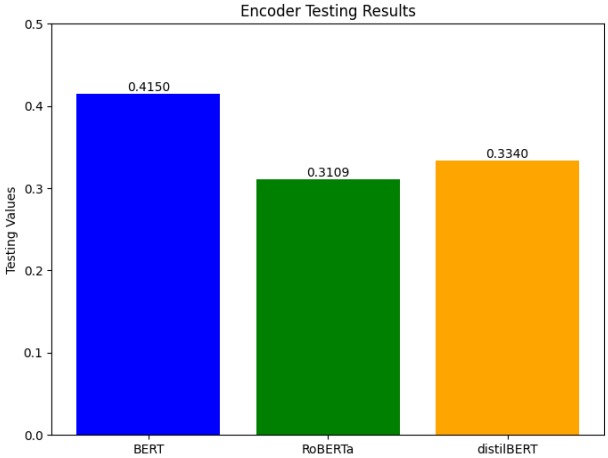


Fig. 5.2 Encoder Training Results

Chapter 6

Conclusion and Future Work

This report presents a chatbot framework designed to incorporate temporal commonsense reasoning, addressing the limitations of existing systems in handling time-sensitive contexts. The framework leverages a combination of temporal and non-temporal models, offering a structured approach to generating contextually accurate and temporally aware responses.

The development of a robust classifier successfully differentiates between temporal and non-temporal contexts, forming the foundation of the framework. The temporal model's encoder, trained using event extraction techniques, demonstrates the ability to associate events with their corresponding temporal attributes, such as timepoints and durations.

Despite these advancements, several challenges and areas for improvement remain, as outlined below.

6.1 Future Work

To further enhance the system's capabilities and generalizability, the following milestones have been identified:

1. **Expansion of Timeline with Additional Datasets:** Expanding the timeline to

cover a wider variety of datasets which include TIMEDIAL as well as other conversational datasets. datasets can enthuse a more diverse base of temporal contexts and consequently outstrip the model's flexibility in relating the ideas in one context to those within another different context.

2. **Integration of Timeline with Decoder:** Integrating timeline data directly into the decoder is expected to enable more contextually accurate temporal reasoning and response generation, enhancing the model's ability to address complex temporal queries.

3. **Development of a Generalized Event Linking Algorithm:** The current event extraction and linking processes respect that heuristics are dataset-dependent. Developing a generalized algorithm which can then be adapted to a range of conversational contexts and connected method for identifying event correlations across multiple datasets is needed to practically apply the concept of event identification at a large scale.

4. **Refinement of Temporal Reasoning:** Improving the model's ability to reason about complex temporal relationships, including overlapping events, conditional dependencies, and sequences with ambiguous timing, will further enhance its temporal understanding.

5. **Evaluation on Unseen Data:** Rigorous evaluation on unseen data is needed in order to define the scenarios where the system usage is most likely to be observed and improve the performance, so that it would not fail in unexpected situations and also effective in terms of application in practice.

In conclusion, the proposed framework establishes a foundation for incorporating temporal commonsense reasoning into conversational systems. By addressing the outlined future

directions, the framework can be further developed into a robust and adaptable system capable of handling diverse and complex temporal contexts effectively.

Chapter 7

References

1. Do Language Models Have a Common Sense regarding Time? Revisiting Temporal Commonsense Reasoning in the Era of Large Language Models by Jain, Raghav and Sojitra, Daivik and Acharya, Arkadeep and Saha, Sriparna and Jatowt, Adam and Dandapat, Sandipan

2. LTLBench: Towards Benchmarks for Evaluating Temporal Logic Reasoning in Large Language Models by Weizhi Tang and Vaishak Belle

3. Toward Building a Language Model for Understanding Temporal Commonsense by Kimura, Mayuko and Kanashiro Pereira, Lis and Kobayashi, Ichiro

4. Formulation Comparison for Timeline Construction using LLMs by Kimihiro Hasegawa and Nikhil Kandukuri and Susan Holm and Yukari Yamakawa and Teruko Mitamura

5. "Going on a vacation" takes longer than "Going for a walk": A Study of Temporal Commonsense Understanding by Ben Zhou, Daniel Khashabi, Qiang Ning, Dan Roth

6. TIMEDIAL : Temporal Commonsense Reasoning in Dialog by Lianhui Qin, Luheng He, Aditya Gupta, Shyam Upadhyay, Yejin Choi, Manaal Faruqui